Python Package evo_tri Documentation

Author: Reed Harder

The evo_tri package contains various functions for implementing the evolutionary triangulation algorithm. These functions carry the user through data setup and analysis from downloading or loading files from HapMap or local sources, to parsing files and calculating pairwise Fsts, to running evolutionary triangulation algorithm, to finding genes in vicinty of SNPs found by evolutionary triangulation algorithm, to graphing number of SNPs and genes found over ranges of Fst cutoffs.

This package requires numpy (and matplotlib for hit_plotter2D() and hitplotter3D())

The evolutionary triangulation algorithm finds pairwise Fsts for shared SNPs between each pairing of three different populations, filters these SNPs for each population pair based on user-selected Fsts cutoffs (for example, SNPs with Fst values >=.45), and then finds the overlapping set of these filtered SNPs between the three populations of interest. Further analysis can be done once this set of overlapping SNPs has been found: genes in the vicinity of these SNPs can be found, and the number of SNPs found with a certain cutoff can be compared to a range of other Fst cuttoffs.

First, population data must be loaded and set up. The evo_tri package contains functions that allow the user to download relevant HapMap allele frequency files, use predownloaded HapMap allele frequency files, or load pre-processed custom population data or precalulated Fst values.

---

evo_tri.evo_hapmapDLer(phase='2009-01_phaseIII', pops=['CEU','YRI','GIH'], chrs='autosome,X', file_folder="current")

```
evo_hapmapDLer: a function for externally downloading hapmap allele_freq files for
an arbitrary number of populations

   parameters:
      phase: String that specifies hapmap phase, e.g. '2009-01_phaseIII'
      pops: list of populations as 3 letter hapmap abbreviations, e.g. 'CEU', or
            'YRI'
      chrs: Enter string of desired chromosomes (1-22,X,Y,M) to be downloaded for
         all populations seperated by commas, no spaces.
         Use "autosome" to select all autosomal chromosomes (e.g.
         "autosome,X,Y"). Use colons to indicate ranges (e.g. "1,4:10,15,Y")
      file_folder: output folder pathname in which to dump downloaded files. Default
        to current directory
```

---

evo_tri.evo_text2numpy(pop1="CEU", pop2="YRI", pop3="GIH", customFst=False, no_popdata=False, pop1filename='pop1textdata.txt', pop2filename='pop2textdata.txt', pop3filename='pop3textdata.txt', pop12filename='pop12textdata.txt', pop13filename='pop13textdata.txt', pop23filename='pop23textdata.txt', file_folder="current", out_compressed=False, pop_out="pop_archive1", FST_out="FST_archive1")

```
evo_text2numpy: converts custom data in text format into a .npz (numpy archive) file,
for efficient processing

parameters:
    pop1,pop2,pop3: hapmap population 3 letter abbrev, or custom population
        abbreviation.
    customFst: True if custom FST data will be provided
    no_popdata: if True, no population data file will be created
    pop[1-3]filenames: text file name strings for each population to be combined in
        numpy array archive. Rows  of text file should be each SNP, columns
        should be rs number, chromosome number, snp position, allele frequency, and
        sample size, in that order, seperated by white space. If certain data not
        required for calculation (for example, if custom FST values will be provided),
        fill in place holder column with -1's.

        SNPs given do not need to be completely overlapping.
    pop[12-13-23]FSTfilenames: text file name strings for each population to be
        combined in numpy array archive. Rows should be each SNP, columns should be
        rs number and associated Fst, seperated by white space. SNPs given do not
        need to be completely overlapping.
    file_folder: input and output folder pathname.
    out_compressed: output file will be compressed
    pop_out: file name (without .npz extension) of population raw data
    FST_out: file name (without .npz extension) of population FST data
```

---

evo_tri.evo_tri_data(datasource=2,
data3customFst=False, no_popdata=False,
custom_popdata='pop_archive1.npz',
custom_FSTdata='FST_archive1.npz',
pop1="CEU", pop2="YRI", pop3="GIH", chrs="autosome,X",
unbiasedFst=True,
phase='2009-01_phaseIII', pop_out="pop_data1",
FST_out="fst_data1", out_compressed=False,
file_folder="current")

```
evo_tri_data: accesses and sets up data, calculates pairwise Fst values for overlaping
SNPs. Saves a dictionary of 2d numpy arrays for each population combo with columns
as rs numbers and corresponding Fst for each SNP common to all three populations.
Also optionally saves a dictionary of 2d numpy arrays for each population with columns
as rs number, chromosome number (X->23,Y->24,M->25), snp position, allele frequency,
and sample size, in that order, for each SNP common to all three populations. See
parameters for details.

parameters:
    datasource: 1 is hapmap online, 2 is local hapmap txt allele_freq text files,
        3 is local custom data (see below for required format), with or without Fst
        and gene location data.
    data3customFst: Only relevant if data source 3 is selected. If True, will use
        custom calculated Fst values, see below. Else, will calculate Fst from given
        data.
    no_popdata: Only relevant if data source 3 is selected and custom Fst will be
          used.

         If true, only SNP/precalculated Fst is taken, allowing for faster
         calculation. Genefinding will not be possible if this option is used.
         custom_popdata, custom_FSTdata: Only relevant if data source 3 is selected.
         File names for preprocessed custom data in .npz files. To convert text data
         files to .npz, see evo_text2numpy(). custom_popdata file should be .npz
         dicionary of 3 numpy matrices of floats, one for each population, keyed with
         '[population abbreviation]'. rows should be SNPs, columns should be rs
         number, chromosome number, snp position, allele frequency, and sample size,
         in that order. Chromosomes X,Y and mitochondrial (M) should be coded as 23,
         24 and 25 respectively. If certain data  are not required for calculation
         (for example, if custom FST values will be provided), fill in place holder
         column with -1's. SNPs given do not need to be completely overlapping.
         If it is being used, custom_FSTdata file should be be .npz dicionary of 3
         numpy matrices of floats, one for each population combo (1-2, 1-3, 2-3),
         keyed with '[population 1 abbreviation] + '[population 2 abbreviation]'.
         Rows should be each SNP, columns should be rs number and
         associated Fst, SNPs given do not need to be completely overlapping.

                 Data required:
                 If custom Fst values are provided, only Fsts and matching rs-number
                 file are required.
                 If unbiased Fst will be calculated, at least rs-numbers, allele
                  frequencies, and sample sizes for each population are required.
                 If uncorrected Fst will be calculated, only rs numbers and allele
                 frequencies are required.
                 If genefinding will be used, chromosome and snp location data for
                 each population is required.
                 Each .txt file should be a column of numerical values corresponding
                 to the order of rs numbers for that population provided.
    pop1,pop2,pop3: hapmap population 3 letter abbrev, or custom population
         abbreviation.
    chrs: Only relevant with datasource 1 and datasource 2. Enter string of desired
         chromosomes (1-22,X,Y,M) seperated by commas, no spaces. Use "autosome" to
         select all autosomal chromosomes (e.g. "autosome,X,Y"). Use colons to
         indicate ranges (e.g. "1,4:10,15,Y")
    phase: only relevant if hapmap files will be downloaded (i.e. datasource 1). String
       that specifies hapmap phase, e.g. '2009-01_phaseIII'
    pop_out, FST_out: string with output file name (.npz extension not necessary).
       For raw population data and calculated Fst data respectively.
    out_compressed: if true, output files are compressed
    file_folder: directory with files, for datasource 2 and 3; directory for storing
       downloaded file database for datasource 1. Defaults to current directory.
```

Once the data has been properly set up and Fst values have been calculated (or provided), the evolutionary traingulation algorithm may be run.

---

evo_tri.evo_triangulator(fst_file='fst_data1.npz', pops=['CEU','YRI','GIH'], pop12lim=">=.45", ptile_cutoff=False, pop13lim=">=.45", pop23lim="<=.05", snps2screen=True, snps2txt=False, snps_filename="evotri_snps", file_folder="current")

```
evo_triangulator(): implements evolutionary triangulation algorithm, saving SNPs
found to .npy and (optionally) text files.

parameters:
    fst_file: name of dictionary containing Fst data for pairings of three populations
          (such as Fst_out from evo_tri_data() )
    pops: list of population abbreviations for population 1, population 2, and
           population 3, in that order
    ptile_cutoff: if True, will consider cuttoffs entered below as percentiles (in
         decimal form, e.g. 95% = .95) rather than absolute cutoffs.
    pop12lim,pop13lim,pop23lim: Fst threshold (string with operator [>,>,>=,<=]
           followed by Fst between 0 and 1) for pop1-pop2 Fsts, pop1-pop3 Fsts, and
           pop2-pop3 Fsts respectively
    snps2screen: true prints snps found to screen
    snps2txt: true prints snps to txt file
    snps_filename: name for snps txt file and .npy file
    file_folder: directory from which to load and save files. Defaults to current
    Directory.
```

Once the evolutionary triangulation algorithm has been run, futher analysis may be performed using the following functions.

---

evo_tri.genefind_local(snplist='evotri_snps.npy', custom_loc=False, loc_data='pop_data1', pop='first', bp_range=100000, custom_gene=False, custom_genefile='custom_genes.txt', genes2txt=False, genes2screen=True, genes_filename='genes1', display_chr=False, file_folder="current")

```
genefind_local: finds genes in regions of overlapping SNPs found (within specified
range) using local gene location data

parameters:
    snplist: .npy file storing array of overlapping SNPs
```

custom_loc: if True, uses .npy file storing 3 column 2d numpy array, with rs numbers, corresponding chromosomes, and corresponding locations, in that order, seperated by white space. Allows use of custom location file, not generated by evo_tri_data.
loc_data: string with .npy or .npz file name (without extension), with file containing SNP location data. If custom_loc==True, provide 3 column file (see above). Otherwise, use .npz file in format of pop_out from evo_tri_data().
pop: only relevant if custom_loc is false. Specify popuation in loc_data from which to take SNP locations. If population is not given or not found, popuation data will be taken from first population in loc_data.
bp_range: integer specifing how many bases from a SNP a gene must be to be considered a hit.
custom_gene: if True, use custom gene text file. File should contain 4 columns seperated by whitespace: chromosome number, gene start, gene end, gene name. Chromosomes X,Y and mitochondrial (M) should be coded as 23, 24 and 25 respectively. If False, will use default genes on file.
genes2text: if True, will save text file of genes found
genes2screen: if True, will print genes found to screen
display_chr: if True, will display and save chromosome number with genes
genes_filename: name for .npy and .txt (if requested) file, where gene data is saved.
file_folder: folder from which to save and load files. Defaults to current directory

---

genefind_ncbi(snplist=[123434,12343557,2342342],bp_range=100000, data_verbose=True, gene2screen=True, file_folder="current", genefile="geneDF.p")

genefind_ncbi: function for getting genes in the vicinity of a list of snps, such as those generated by evo_triangulator, using ncbi gene databases
parameters:
snplist: numpy array or list of snp numbers to find nearby genes for
bp_range: range of base pairs on either side of snp in which to search for genes
data_verbose: if True, list of genes for each snp will be a list of dictionaries of various additional gene data:
gene name, chromosome, description, aliases, gene start position, gene stop position,summary
keyed as, respectively:
'Name','Chr','Description','Alias','Start','Stop','Summary'
gene2screen: if True, will print summary of genes found to screen
file_folder: directory in which to save gene dataframes
genefile: file name to save dataframe
returns:
with data_verbose == True: a list of two dataFrames, first one with simple lists of genes as final entry for each snp, second with lists of dictionaries with gene information for each snp (as described above)
with data_verbose == False: just returns simple data frame

---

evo_tri.hit_plotter2D(backend="TkAgg", granularity=10, fst_file='fst_data1.npz', pops=['CEU','YRI','GIH'], plot_snps=True, plot_genes=True, pop12lim=">=X", pop13lim=">=.45", pop23lim="<=.05", loc_data='pop_data1', bp_range=100000, file_folder="current")

hit_plotter2D: function to plot number of hits (genes and/or snps) that the evolutionary triangulator finds against a changing Fst on one
of the population-pair axes

parameters:
  backend: enter string to select matplotlib backend for generating figure
    granularity: how many points to plot between an Fst of 0 and 1
  fst_file: Fst data to perform evolutionary tringulation data on. Should be a name of dictionary containing Fst data for pairings of three populations (such as Fst_out from evo_tri_data())
  pops: list of population abbreviations for population 1, population 2, and population 3, in that order
  pop12lim,pop13lim,pop23lim: Fst threshold (string with operator [>,>,>=,<=] followed by Fst between 0 and 1) for pop1-pop2 Fsts, pop1-pop3 Fsts, and pop2-pop3 Fsts respectively
    One and only one of these thresholds should be an operator followed by character X. This will be the axis along which the value changes.
    loc_data: if plotting gene hits, provide a string with .npy or .npz file name (without extension), with file containing SNP location data, in format of pop_out from evo_tri_data().
  bp_range: if plotting gene hits, provide an integer specifing how many bases from a SNP a gene must be to be considered a hit.
  file_folder: folder from which to save and load files. Default to current directory

---

evo_tri.hit_plotter3D(backend="TkAgg", granularity=10, fst_file='fst_data1.npz', pops=['CEU','YRI','GIH'], plot_snps=True, plot_genes=True, pop12lim=">=X", pop13lim=">=Y", pop23lim="<=.05", gene_maxZ=100, snp_maxZ=300, loc_data='pop_data1', bp_range=100000, file_folder="current")

hit_plotter3D: function to plot number of hits (genes and/or snps) that the evolutionary triangulator finds against a changing Fst on two
of the population-pair axes. This function can take a long time to run, depending on granularity.

parameters:
  backend: enter string to select matplotlib backend for generating figure
    granularity: how many points to plot between an Fst of 0 and 1
  fst_file: Fst data to perform evolutionary tringulation data on. Should be a name of dictionary containing Fst data for pairings of three populations (such as Fst_out from evo_tri_data())
  pops: list of population abbreviations for population 1, population 2, and population 3, in that order
  pop12lim,pop13lim,pop23lim: Fst threshold (string with operator [>,>,>=,<=]

```
    followed by Fst between 0 and 1) for pop1-pop2 Fsts, pop1-pop3 Fsts, and
    pop2-pop3 Fsts respectively. One of these thresholds should be an operator
    followed by character X. This will be the first axis along which the value
    Changes. Another of these thresholds shoul be an operator followed by character
    Y. This will be the second axis along which the value changes.
loc_data: if plotting gene hits, provide a string with .npy or .npz file name
    (without extension), with file containing SNP location data, in format of
    pop_out from evo_tri_data().
bp_range: if plotting gene hits, provide an integer specifing how many bases from
    a SNP a gene must be to be considered a hit.
gene_maxZ: max of z-axis range for number of genes
snp_maxZ: max of z-axis range for number of genes
file_folder: folder from which to save and load files. Defaults to current
diectory.
```